Aprendendo python - Unidade 1

Primeiros passos com python

In [ ]:
```
print("hello world!!!")
```

```
hello world!!!
```

variáveis e tipos básicos em python

In [ ]:
```
x = 10 #inteiro
nome = 'aluno' #caractere
nota = 8.75 #float
fez_inscricao = True #boleano
```

In [ ]:
```
print(type(x))
print(type(nome))
print(type(nota))
print(type(fez_inscricao))
```

```
<class 'int'>
<class 'str'>
<class 'float'>
<class 'bool'>
```

In [ ]:
```
nome = input("Digite o seu nome: ")
print(nome)
```

```
Ryan
```

Em python, temos uma variedade de formas de imprimir texto com variaveis. podemos usar formatadores de caracteres como na linguagem C, usar a função format(), ou criar uma string formatada, veja exemplos.

In [ ]:
```
# Modo 1 - usando formatadores de caracteres (igual na linguagem C) para imprimi
print("Olá %s, bem vindo a disciplina de programação. Parabéns pelo seu primeiro

# Modo 2 - usando a função format() para imprimir variável e texto
print("Olá {}, bem vindo a disciplina de programação. Parabéns pelo seu primeiro

# Modo 3 - usando strings formatadas
print(f"Olá {nome}, bem vindo a disciplina de programação. Parabéns pelo seu pri
```

```
Olá Ryan, bem vindo a disciplina de programação. Parabéns pelo seu primeiro hel
lo world
Olá Ryan, bem vindo a disciplina de programação. Parabéns pelo seu primeiro hel
lo world
Olá Ryan, bem vindo a disciplina de programação. Parabéns pelo seu primeiro hel
lo world
```

Operações matemáticas em python

Operação

Resultado

x + y

soma de x e y

x - y

Diferença de x e y

x * y

Produto de x e y

x / y

Quociente de x e y

x // y

Parte inteira do quociente de x e y

x % y

Resto de x / y

abs(x)

Valor absoluto de x

pow(x, y)

x elevado a y

x ** y

x elevado a y

In [ ]:
```
# Qual o resultado armazendo na variável operacao_1: 25 ou 17?
operacao_1 = 2 + 3 * 5

# Qual o resultado armazendo na variável operacao_2: 25 ou 17?
operacao_2 = (2 + 3) * 5

# Qual o resultado armazendo na variável operacao_3: 4 ou 1?
operacao_3 = 4 / 2 ** 2

# Qual o resultado armazendo na variável operacao_4: 1 ou 5?
operacao_4 = 13 % 3 + 4

print(f"Resultado em operacao_1 = {operacao_1}")
print(f"Resultado em operacao_2 = {operacao_2}")
print(f"Resultado em operacao_3 = {operacao_3}")
print(f"Resultado em operacao_4 = {operacao_4}")
```

```
Resultado em operacao_1 = 17
Resultado em operacao_2 = 25
Resultado em operacao_3 = 1.0
Resultado em operacao_4 = 5
```

Desafio

In [ ]:
```
primeiro_mes = 200
#segundo mes = 400
#terceiro mes = 600
#quarto mes = 800
#quinto mes = 1000
#sexto mes = 1200

mes_previsao = int(input("Digite o número correspondente ao mês que voçê quer sa
previsao = mes_previsao * primeiro_mes
print(f"O número de vendas previsto no mes {mes_previsao} é de {previsao} peças!
```

Estruturas lógicas, condicionais e de repetição em python

IF, ELIF E ELSE

In [ ]:
```
a = int(input("Digite o valor de A: "))
b = int(input("Digite o valor de B: "))

if a < b:
    print(f"{a} é menor que {b}")
else:
    print(f"{a} não é menor que {b}")
```

```
2 é menor que 5
```

In [ ]:
```
codigo_compra = 5111

if codigo_compra == 5111:
    print("Compra a vista!")
elif codigo_compra == 5222:
    print("Compra a prazo no cartao!")
elif codigo_compra == 5333:
    print("Compra a prazo no boleto!")
else:
    print("Código não cadastrado")
```

```
Compra a vista!
```

AND, OR, NOT

Além dos operadores relacionais, podemos usar os operadores booleanos para construir estruturas de decisões mais complexas.

Operador booleano and: Esse operador faz a operação lógica E, ou seja, dada a expressão (a and b), o resultado será True, somente quando os dois argumentos forem verdadeiros.

Operador booleano or: Esse operador faz a operação lógica OU, ou seja, dada a expressão (a or b), o resultado será True, quando pelo menos um dos argumentos for verdadeiro.

Operador booleano not: Esse operador faz a operação lógica NOT, ou seja, dada a expressão (not a), ele irá inverter o valor do argumento. Portanto, se o argumento for verdadeiro, a operação o transformará em falso e vice-versa.

In [ ]:
```
qtde_faltas = int(input("Digite a quantidade de faltas: "))
media_final = float(input("Digite a média final: "))

if qtde_faltas <= 5 and media_final >= 6:
    print("Aprovado!")
elif qtde_faltas <= 5 and media_final < 6:
    print("Recuperação!")
else:
    print("Reprovado!")
```

```
Aprovado!
```

WHILE E FOR

O comando while deve ser utilizado para construir e controlar a estrutura decisão, sempre que o número de repetições não seja conhecido.

In [ ]:
```
numero = 1
while numero != 0:
    numero = int(input("Digite um numero: "))
    if numero % 2 == 0:
        print("Numero par!")
    else:
        print("Numero impar!")
```

```
Numero par!
Numero impar!
Numero par!
```

Outro comando muito utilizado para construir as estruturas de repetição é o for. A instrução for em Python difere um pouco do que ocorre em outras linguagens, como C ou Java, com as quais você pode estar acostumado.

In [ ]:
```
nome = "ryan"
for c in nome:
    print(c)
```

```
r
y
a
n
```

In [ ]:
```
nome = "carlos"
for index, c in enumerate(nome):
    print(f"posicao: {index}, valor: {c}")
```

```
posicao: 0, valor: c
posicao: 1, valor: a
posicao: 2, valor: r
posicao: 3, valor: l
posicao: 4, valor: o
posicao: 5, valor: s
```

RANGE, BREAK E CONTINUE

Para criar uma sequência numérica de iteração em Python, podemos usar a função range().

Método 1: passando um único argumento que representa a quantidade de vezes que o laço deve repetir;

Método 2: passando dois argumentos, um que representa o início das repetições e outro o limite superior (NÃO INCLUÍDO) do valor da variável de controle;

Método 3: Passando três argumentos, um que representa o início das repetições; outro, o limite superior (NÃO INCLUÍDO) do valor da variável de controle e um que representa o incremento. Observe as três maneiras a seguir.

In [ ]:
```
for x in range(0,100, 5):
    print(x)
```

```
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
```

Além de controlar as iterações com o tamanho da sequência, outra forma de influenciar no fluxo é por meio dos comandos "break" e "continue". O comando break para a execução de uma estrutura de repetição, já com o comando continue, conseguimos "pular" algumas execuções, dependendo de uma condição.

In [ ]:
```
#exemplo de uso do break
disciplina = "Linguagem de programação"

for c in disciplina:
    if c == 'a':
```

```
        break
    else:
        print(c)
```

```
L
i
n
g
u
```

In [ ]:
```
#exemplo de uso do continue
for c in disciplina:
    if c == 'a':
        continue
    else:
        print(c)
```

```
L
i
n
g
u
g
e
m

d
e

p
r
o
g
r
m
ç
ã
o
```

In [ ]:
```
texto = """
A inserção de comentários no código do programa é uma prática normal.
Em função disso, toda linguagem de programação tem alguma maneira de permitir qu
O objetivo é adicionar descrições em partes do código, seja para documentá-lo ou
"""
for i, c in enumerate(texto):
    if c == 'a' or c == 'e':
        print(f"Vogal '{c}' encontrada, na posição {i}")
    else:
        continue
```

```
Vogal 'e' encontrada, na posição 6
Vogal 'e' encontrada, na posição 13
Vogal 'e' encontrada, na posição 18
Vogal 'a' encontrada, na posição 45
Vogal 'a' encontrada, na posição 47
Vogal 'a' encontrada, na posição 53
Vogal 'a' encontrada, na posição 61
Vogal 'a' encontrada, na posição 67
Vogal 'a' encontrada, na posição 91
Vogal 'a' encontrada, na posição 98
Vogal 'e' encontrada, na posição 100
Vogal 'e' encontrada, na posição 104
Vogal 'a' encontrada, na posição 111
Vogal 'a' encontrada, na posição 113
Vogal 'e' encontrada, na posição 119
Vogal 'a' encontrada, na posição 122
Vogal 'a' encontrada, na posição 127
Vogal 'a' encontrada, na posição 130
Vogal 'e' encontrada, na posição 132
Vogal 'a' encontrada, na posição 135
Vogal 'e' encontrada, na posição 138
Vogal 'e' encontrada, na posição 141
Vogal 'e' encontrada, na posição 151
Vogal 'e' encontrada, na posição 156
Vogal 'e' encontrada, na posição 166
Vogal 'a' encontrada, na posição 168
Vogal 'e' encontrada, na posição 174
Vogal 'a' encontrada, na posição 190
Vogal 'a' encontrada, na posição 192
Vogal 'e' encontrada, na posição 201
Vogal 'a' encontrada, na posição 209
Vogal 'a' encontrada, na posição 216
Vogal 'e' encontrada, na posição 220
Vogal 'e' encontrada, na posição 227
Vogal 'e' encontrada, na posição 230
Vogal 'a' encontrada, na posição 234
Vogal 'e' encontrada, na posição 237
Vogal 'e' encontrada, na posição 252
Vogal 'a' encontrada, na posição 254
Vogal 'a' encontrada, na posição 257
Vogal 'a' encontrada, na posição 259
Vogal 'e' encontrada, na posição 266
Vogal 'a' encontrada, na posição 278
Vogal 'a' encontrada, na posição 280
Vogal 'a' encontrada, na posição 282
Vogal 'a' encontrada, na posição 289
Vogal 'a' encontrada, na posição 294
Vogal 'e' encontrada, na posição 297
Vogal 'a' encontrada, na posição 309
Vogal 'e' encontrada, na posição 323
Vogal 'e' encontrada, na posição 325
Vogal 'a' encontrada, na posição 328
```

Desafio

Dando continuidade ao seu trabalho na empresa de consultoria de software, o cliente que fabrica peças automotivas requisitou uma nova funcionalidade para o sistema: calcular o imposto de renda a ser deduzido do salário dos colaboradores. O imposto de renda "Incide sobre a renda e os proventos de contribuintes residentes no País ou

residentes no exterior que recebam rendimentos de fontes no Brasil. Apresenta alíquotas variáveis conforme a renda dos contribuintes, de forma que os de menor renda não sejam alcançados pela tributação." (IRPF, 2020)

Você foi designado para pesquisar a tabela de valores para o imposto de renda do ano de 2020, pensar no algoritmo e implementar a primeira versão da solução. Quando o cliente aprovar, a versão passará para o time de produção. Nessa primeira versão, o programa deve solicitar o salário do colaborador e então, informar qual o valor do imposto que será deduzido do salário.

Base de cálculo do Imposto de Renda		
Base de cálculo	**Alíquota**	**Parcela a deduzir**
Até R$1.903,98*	Isento	Isento
De R$1.903,99 até R$2.826,65	7,5%	R$ 142,80
De R$2.826,66 até R$3.751,05	15%	R$ 354,80
De R$3.751,06 até R$4.664,68	22,5%	R$ 636,13
Acima de R$4.664,68	27,5%	R$ 869,36

In [ ]:
```
def verifica_salario(salario):
    aliquota = 'isento'
    deducao = 'isento'

    if salario <= 1903.98:
        print(f"Aliquota = {aliquota} , Parcela a deduzir: {deducao}")

    elif salario >= 1903.99 and salario <= 2826.65:
        aliquota = float(7.5)
        deducao = float(142.80)
        print(f"Aliquota = {aliquota}% , Parcela a deduzir: R$ {deducao}")

    elif salario >= 2826.66 and salario <= 3751.05:
        aliquota = float(15)
        deducao = float(354.80)
        print(f"Aliquota = {aliquota}% , Parcela a deduzir: R$ {deducao}")

    elif salario >= 3751.06 and salario <= 4664.68:
        aliquota = float(22.5)
        deducao = float(636.13)
        print(f"Aliquota = {aliquota}% , Parcela a deduzir: R$ {deducao}")
    else:
        aliquota = float(27.5)
        deducao = float(869.36)
        print(f"Aliquota = {aliquota}% , Parcela a deduzir: R$ {deducao}")


def pegar_dados_usuario():
    salario_usuario = float(input("Digite o seu salário: "))
    verifica_salario(salario_usuario)
```

```
decisao = 1
while (decisao == 1):
    pegar_dados_usuario()
    decisao = int(input("Digite 1 para continuar e 0 para sair"))
```

```
Aliquota = 22.5% , Parcela a deduzir: R$ 636.13
Aliquota = 27.5% , Parcela a deduzir: R$ 869.36
```

Funções em python

IMPLEMENTANDO SOLUÇÕES EM PYTHON MEDIANTE FUNÇÕES

FUNÇÕES BUILT-IN EM PYTHON

Desde que escrevemos nossa primeira linha de código nessa disciplina >>print("hello world"), já começamos a usar funções, pois print() é uma função built-in do interpretador Python. Uma função built-in é um objeto que está integrado ao núcleo do interpretador, ou seja, não precisa ser feita nenhuma instalação adicional, já está pronto para uso. O interpretador Python possui várias funções disponíveis

- abs()
- delattr()
- hash()
- memoryview()
- set()
- all()
- dict()
- help()
- min()
- setattr()
- any()
- dir()
- hex()
- next()
- slice()
- ascii()
- divmod()
- id()
- object()
- sorted()
- bin()
- enumerate()
- input()
- oct()
- staticmethod()
- bool()

- eval()
- int()
- open()
- str()
- breakpoint()
- exec()
- isinstance()
- ord()
- sum()
- bytearray()
- filter()
- issubclass()
- pow()
- super()
- bytes()
- float()
- iter()
- print()
- tuple()
- callable()
- format()
- len()
- property()
- type()
- chr()
- frozenset()
- list()
- range()
- vars()
- classmethod()
- getattr()
- locals()
- repr()
- zip()
- compile()
- globals()
- map()
- reversed()
- **import**()
- complex()
- hasattr()
- max()
- round()

Funções criadas

In [ ]:
```
def imprimir_mensagem(disciplina, curso):
    print(f"A sua diciplina é {disciplina} e o seu curso é {curso}")

def pegar_mensagem():
    disciplina = input("Qual o nome da sua disciplina?: ")
    curso = input("Qual o nome do seu curso?: ")
    imprimir_mensagem(disciplina=disciplina, curso=curso)

pegar_mensagem()
imprimir_mensagem('Programação orientada a objetos','Ciência de dados')
```

```
A sua diciplina é Linguagem de programação e o seu curso é ciencia de dados
A sua diciplina é Programação orientada a objetos e o seu curso é Ciência de da
dos
```

In [ ]:
```
#converter mes para extenso

def converter_mes_para_extenso(data):
    mes = '''janeiro fevereiro marco abril maio junho julho agosto setembro outu

    d, m, a = data.split('/')
    mes_extenso = mes[int(m) - 1]
    return f'{d} de {mes_extenso} de {a}'

print(converter_mes_para_extenso('15/07/2002'))
```

```
15 de julho de 2002
```

In [ ]:
```
#parametros indefinidos
def cadastrar_pessoa(nome, idade, cidade):
    print("\nDados a serem cadastrados: ")
    print(f"Nome: {nome}")
    print(f"Idade: {idade}")
    print(f"Cidade: {cidade}")
    print("\n")

cadastrar_pessoa("Joao", 23, "Ipatinga")
cadastrar_pessoa("Ryan", 20, "Coronel Fabriciano")
```

```
Dados a serem cadastrados:
Nome: Joao
Idade: 23
Cidade: Ipatinga



Dados a serem cadastrados:
Nome: Ryan
Idade: 20
Cidade: Coronel Fabriciano
```

In [ ]:
```
def imprimir_parametros(*args):
    qtde_parametros = len(args)
    print(f"Quantidade de parâmetros = {qtde_parametros}")

    for i, valor in enumerate(args):
        print(f"Posição = {i}, valor = {valor}")
```

```
print("\nChamada 1")
imprimir_parametros("São Paulo", 10, 23.78, "João")

print("\nChamada 2")
imprimir_parametros(10, "São Paulo")
```

```
Chamada 1
Quantidade de parâmetros = 4
Posição = 0, valor = São Paulo
Posição = 1, valor = 10
Posição = 2, valor = 23.78
Posição = 3, valor = João

Chamada 2
Quantidade de parâmetros = 2
Posição = 0, valor = 10
Posição = 1, valor = São Paulo
```

FUNÇÕES ANÔNIMAS EM PYTHON

In [ ]:
```
(lambda x, y: x+y)(x=3, y=3)
```

Out[ ]: 6

Desafio

Dando continuidade ao seu trabalho na empresa de consultora de software, o cliente que fabrica peças automotivas requisitou uma nova funcionalidade para o sistema: calcular o total de vendas. Como seus negócios estão expandindo, o cliente solicitou que o sistema seja capaz de receber valores em reais, dolár ou euro, e que seja feita a conversão para o valor em reais. Como essa entrega demanda a construção de uma interface gráfica, além da comunicação com banco de dados, dentre outros requisitos técnicos, você foi alocado em uma equipe ágil para criar a função que fará o cálculo do valor.

Após uma primeira reunião, a equipe fez um levantamento de requisitos e concluiu que a função a ser construída precisa considerar os seguintes itens:

- O valor do produto (obrigatório). - Concluido
- A quantidade do produto (obrigatório). - Concluido
- A moeda em que deve ser feito o cálculo (obrigatório, sendo padrão o real). - Concluido
- A porcentagem do desconto que será concedida na compra (opcional). - Concluido
- A porcentagem de acréscimo, que depende da forma de pagamento (opcional). - Concluido

In [ ]:
```
# Calculo do total de vendas

def converter_valores(valor, tipo_moeda, qtde_produto):
    dolar = 5.20
    euro = 6

    pagamento = int(input(
```

```
        "Forma de pagamento -> \n1 - a vista \n2 - a prazo no boleto \n3 - a pra

    if tipo_moeda == "1":
        valor_real = valor * qtde_produto
        forma_de_pagamento(pagamento=pagamento, valor=valor_real)
    elif tipo_moeda == "2":
        valor_real = (valor * dolar) * qtde_produto
        forma_de_pagamento(pagamento=pagamento, valor=valor_real)
    elif tipo_moeda == "3":
        valor_real = (valor * euro) * qtde_produto
        forma_de_pagamento(pagamento=pagamento, valor=valor_real)
    else:
        print('Valor padrão para moedas : Real')
        valor_real = valor * qtde_produto
        forma_de_pagamento(pagamento=pagamento, valor=valor_real)


def valida_valores(valor, tipo_moeda, qtde_produto):

    valor_produto = valor
    moeda = tipo_moeda
    qtde = qtde_produto

    if valor_produto == "" or moeda == "" or qtde == "":
        print("\nObrigatorio informar o todos os dados!")
    else:
        converter_valores(valor=float(valor_produto),
                          tipo_moeda=int(moeda), qtde_produto=int(qtde))


def forma_de_pagamento(pagamento, valor):
    if pagamento == 1:
        porcentagem = 0.10
        desconto = valor * porcentagem
        valor_a_vista = valor - desconto
        print(
            f"Valor {valor:.2f}, Desconto de {porcentagem * 100}%, valor a pagar
    elif pagamento == 2:
        porcentagem = 0.10
        acrecimo = valor * porcentagem
        valor_no_boleto = valor + acrecimo
        print(
            f"Valor {valor:.2f}, Acrécimo de {porcentagem * 100}%, valor a pagar
    elif pagamento == 3:
        porcentagem = 0.15
        acrecimo = valor * porcentagem
        valor_no_cartao = valor + acrecimo
        print(
            f"Valor {valor:.2f}, Acrécimo de {porcentagem * 100}%, valor a pagar


def pegar_valores():
    valor = input("Valor do produto: ")
    tipo_moeda = input("Tipo de moeda \n1-real \n2-dolar \n3-euro): ")
    qtde_produto = input("Quantidade: ")
    valida_valores(valor=valor, tipo_moeda=tipo_moeda,
                   qtde_produto=qtde_produto)

    while (valor == "" or tipo_moeda == "" or qtde_produto == ""):
        valor = input("Valor do produto: ")
```

```
        tipo_moeda = input("Tipo de moeda \n1-real \n2-dolar \n3-euro): ")
        qtde_produto = input("Quantidade: ")
        valida_valores(valor=valor, tipo_moeda=tipo_moeda,
                       qtde_produto=qtde_produto)


pegar_valores()
```

```
Valor padrão para moedas : Real
Valor 144.00, Desconto de 10.0%, valor a pagar R$129.60, modalidade de pagament
o 1 - A vista
```